# Spring Cloud로 개발하는 마이크로서비스 애플리케이션

# 목차

Part II

Section 16.

# 애플리케이션 배포

# Docker Container

- Design Deployment
- Configuration server
- Eureka Discovery
- API Gateway
- MariaDB
- Kafka
- Zipkin
- Monitoring
- Microservices
- Multiple Environments

# *Running Microservices*

- IntelliJ IDEA
- Exported JAR file
- Docker Container


***IntelliJ IDEA + Local***

***JAR file + Local***

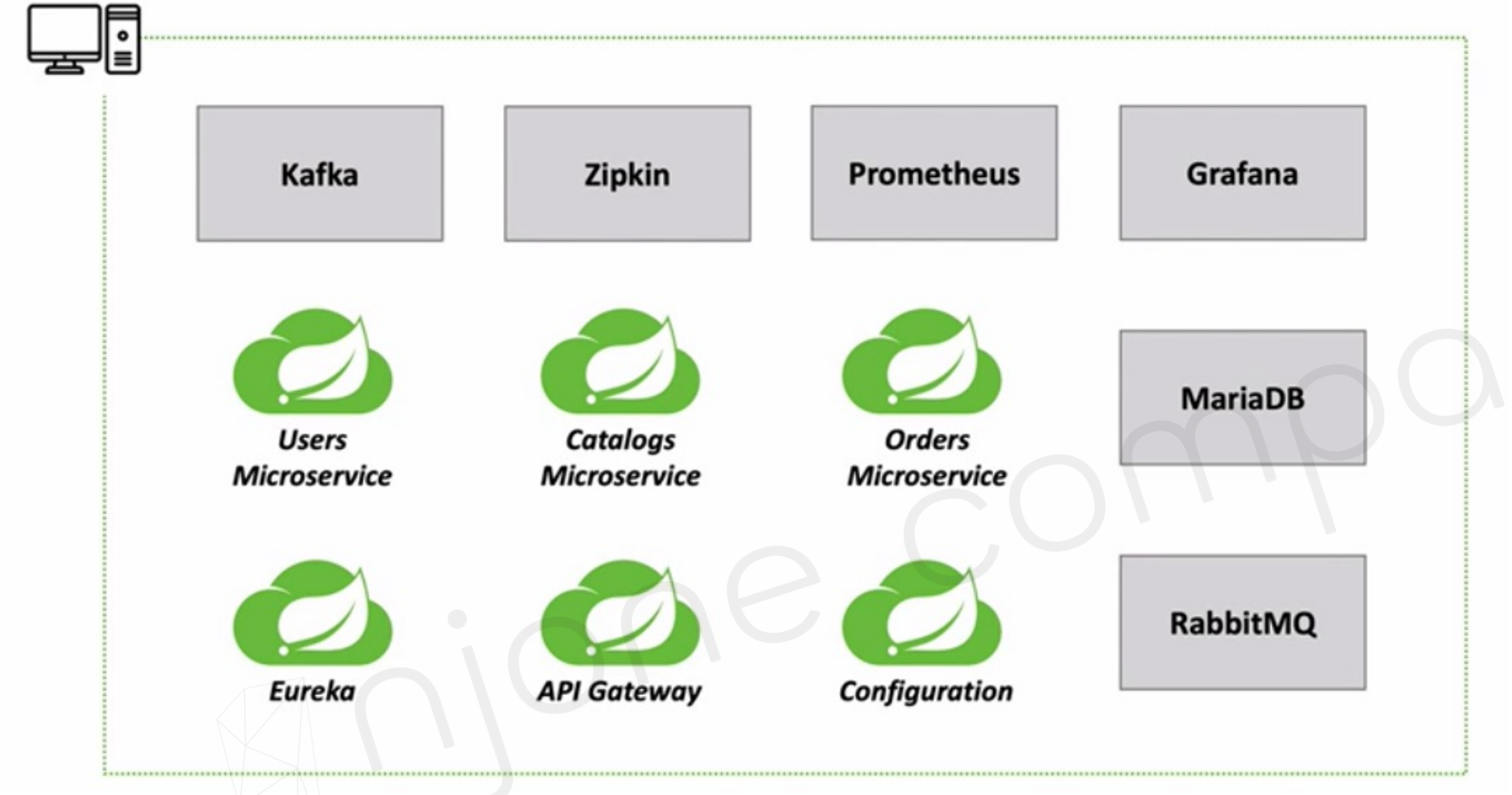
***Docker + Local***

***Docker + AWS EC2***

***Docker + Docker Swarm Mode + AWS EC2***

***Docker + Kubernetes + AWS EC2***

# *Running Microservices in Local*



# *Create Bridge Network*

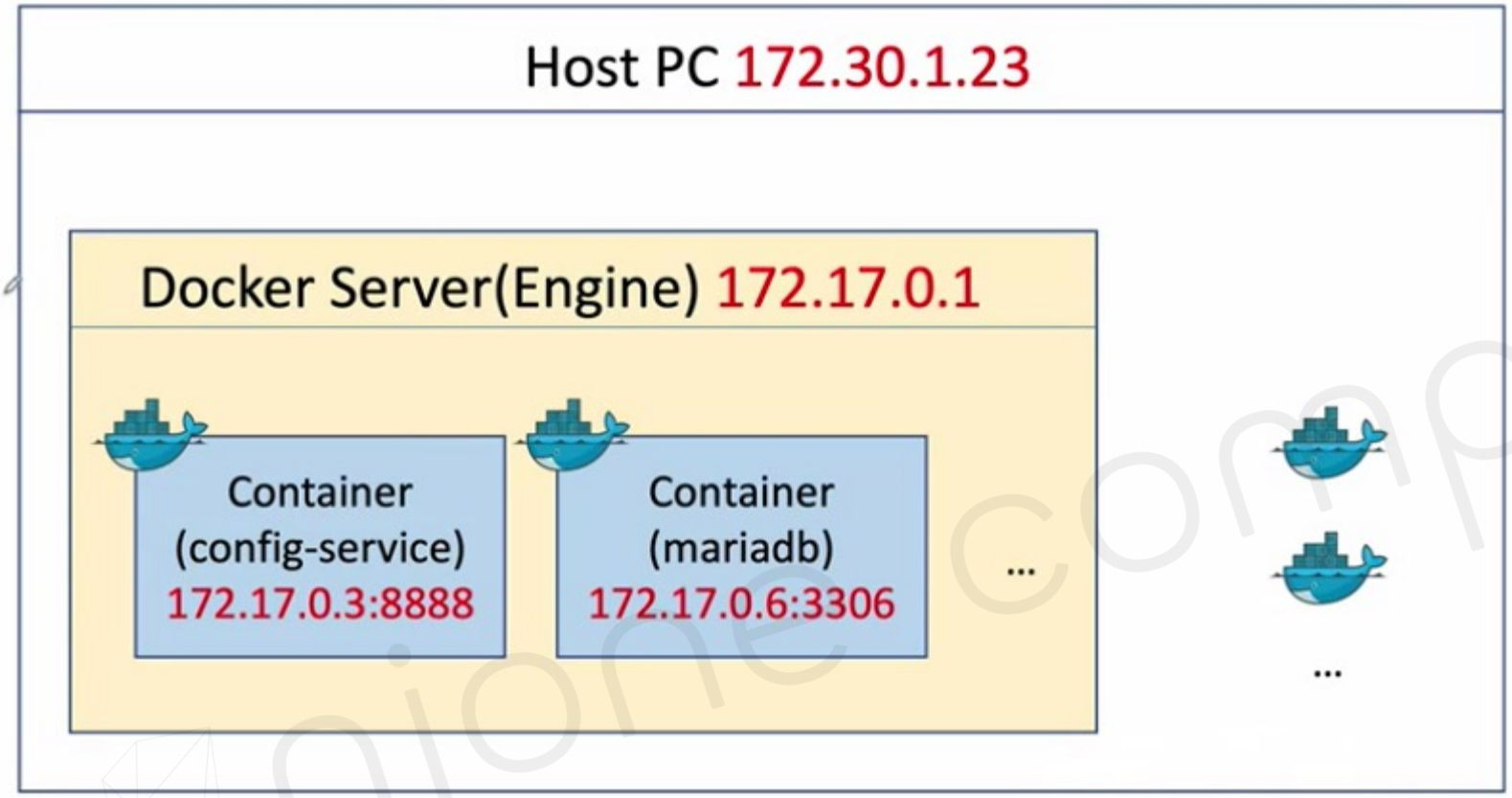
- ***Bridge network***
  - $ docker network create --driver bridge [브릿지 이름]
- ***Host network***
  - 네트워크를 호스트로 설정하면 호스트의 네트워크 환경을 그대로 사용
  - 포트 포워딩 없이 내부 어플리케이션 사용
- ***None network***
  - 네트워크를 사용하지 않음
  - lo 네트워크만 사용, 외부와 단절

$ docker network create ***ecommerce-network***

$ docker network ls

```
docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
2879c2a94c9f   bridge              bridge    local
22e70794e399   ecommerce-network   bridge    local
de014ae20e46   host                host      local
2624444d0133   none                null      local
```

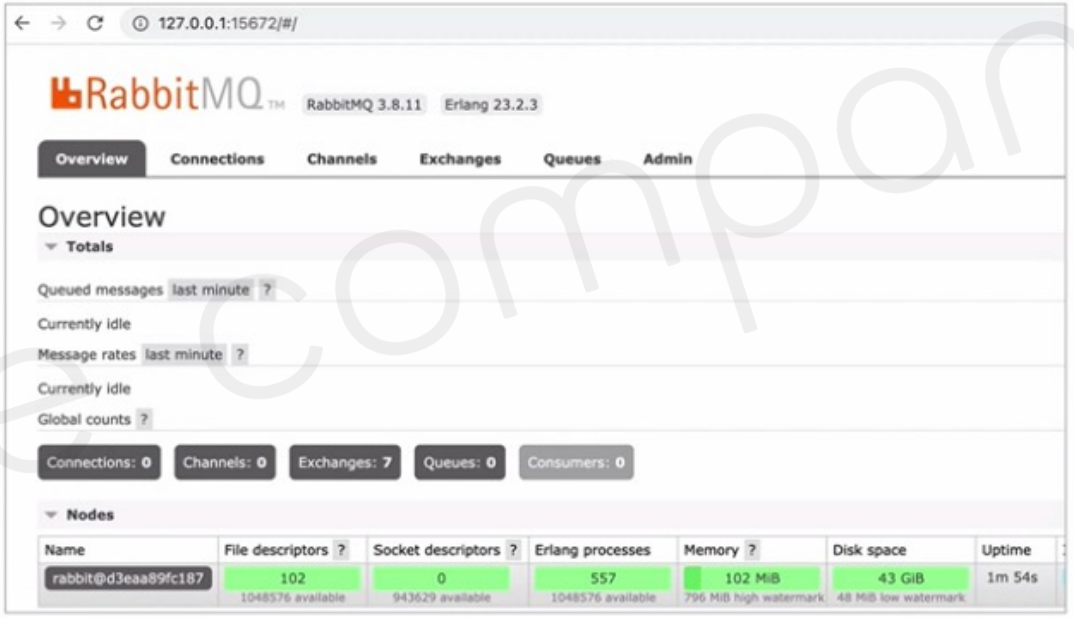
# Create Bridge Network



# *Run RabbitMQ*

```
$ docker run -d --name rabbitmq --network ecommerce-network \
        -p 15672:15672 -p 5672:5672 -p 15671:15671 -p 5671:5671 -p 4369:4369 \
        -e RABBITMQ_DEFAULT_USER=guest \
        -e RABBITMQ_DEFAULT_PASS=guest rabbitmq:management
```

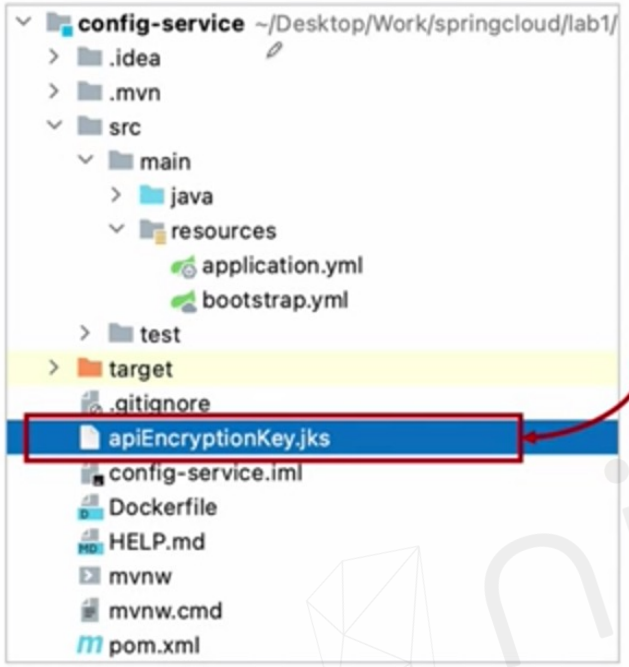
```
Starting RabbitMQ 3.8.11 on Erlang 23.2.3
Copyright (c) 2007-2020 VMware, Inc. or its affiliates.
Licensed under the MPL 2.0. Website: https://rabbitmq.com

 ##  ##      RabbitMQ 3.8.11
 ##  ##
 ##########  Copyright (c) 2007-2020 VMware, Inc. or its affiliates.
 ######  ##
 ##########  Licensed under the MPL 2.0. Website: https://rabbitmq.com

 Doc guides: https://rabbitmq.com/documentation.html
 Support:    https://rabbitmq.com/contact.html
 Tutorials:  https://rabbitmq.com/getstarted.html
 Monitoring: https://rabbitmq.com/monitoring.html
```



# Create Config Server Docker Image

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY apiEncryptionKey.jks apiEncryptionKey.jks
COPY target/config-service-1.0.jar ConfigService.jar
ENTRYPOINT ["java", "-jar", "ConfigService.jar"]
```

```
<groupId>com.example</groupId>
<artifactId>config-service</artifactId>
<!--    <version>0.0.1-SNAPSHOT</version>-->
<version>1.0</version>
<name>config-service</name>
```




- ***copy to docker container***

# Create Config Server Docker Image

```
encrypt:
#  key: abcdefghijklmnopqrstuvwxyz0123456789
  key-store:
#    location: file://${user.home}/Desktop/Work/keystore/apiEncryptionKey.jks
    location: file:/apiEncryptionKey.jks
    password: test1234
    alias: apiEncryptionKey
```

- ***change the path***

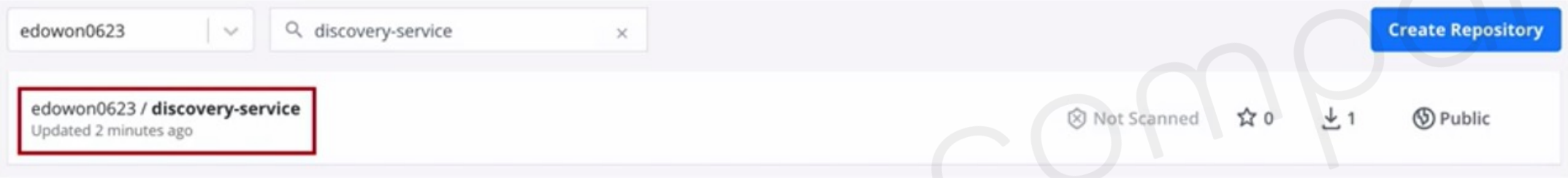
$ mvn clean compile package –DskipTests=true

```
[INFO] --- maven-surefire-plugin:2.22.2:test (default-test) @ config-service ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ config-service ---
[INFO] Building jar: /Users/dowonlee/Desktop/Work/springcloud/lab1/config-service/target/config-service-1.0.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.4.2:repackage (repackage) @ config-service ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.616 s
[INFO] Finished at: 2021-03-08T09:05:27+09:00
[INFO] ------------------------------------------------------------------------
```

# Build Eureka Discovery Docker Image

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY target/discoveryservice-1.0.jar DiscoveryService.jar
ENTRYPOINT ["java", "-jar", "DiscoveryService.jar"]
```

- docker build --tag= edowon0623/discovery-service .
- docker push edowon0623/discovery-service



# Run Eureka Discovery

```
$ docker run -d -p 8761:8761 --network ecommerce-network \
        -e "spring.cloud.config.uri=http://config-service:8888" \
        --name discovery-service edowon0623/discovery-service
```

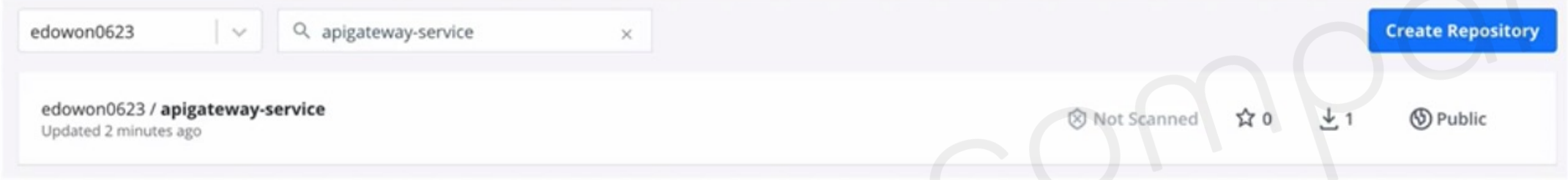
```
$ docker logs discovery-service
```

```
INFO 1 --- [      Thread-10] o.s.c.n.e.server.EurekaServerBootstrap   : isAws returned false
INFO 1 --- [      Thread-10] o.s.c.n.e.server.EurekaServerBootstrap   : Initialized server context
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl    : Got 1 instances from neighboring DS node
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl    : Renew threshold is: 1
INFO 1 --- [      Thread-10] c.n.e.r.PeerAwareInstanceRegistryImpl    : Changing status to UP
INFO 1 --- [      Thread-10] e.s.EurekaServerInitializerConfiguration : Started Eureka Server
INFO 1 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer   : Tomcat started on port(s): 8010 (http) with context path ''
INFO 1 --- [          main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8010
```

# Build ApiGateway Service Docker Image

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY target/apigateway-1.0.jar ApiGateway.jar
ENTRYPOINT ["java", "-jar", "ApiGateway.jar"]
```

- docker build --tag=edowon0623/apigateway-service .
- docker push edowon0623/apigateway-service



# Run ApiGateway Service

```yaml
spring:
  cloud:
    config:
      uri: http://127.0.0.1:8888
      name: ecommerce
```

```yaml
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka
```

```
$ docker run -d -p 8000:8000 --network ecommerce-network \
        -e "spring.cloud.config.uri=http://config-service:8888" \
        -e "spring.rabbitmq.host=rabbitmq" \
        -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" \
        --name apigateway-service \
        edowon0623/apigateway-service
```

# Build, Run MariaDB

```
FROM mariadb
ENV MYSQL_ROOT_PASSWORD test1357
ENV MYSQL_DATABASE mydb
COPY ./mysql /var/lib/mysql
EXPOSE 3306
ENTRYPOINT ["mysqld"]
```

```
mysql.server start
Starting MariaDB
.210308 23:25:30 mysqld_safe Logging to '/usr/local/var/mysql/DOWONui-MacBookPro.local.err'.
210308 23:25:30 mysqld_safe Starting mariadbd daemon with databases from /usr/local/var/mysql
 SUCCESS!
```

```
C:₩Work₩mariadb-10.5.8-winx64>.₩bin₩mariadb-install-db.exe --datadir=C:₩Work₩mariadb-10.5.8-winx64₩data --service=mariaDB --port=3306 --password=test1357
Running bootstrap
2021-02-21 12:45:55 0 [Note] C:₩Work₩mariadb-10.5.8-winx64₩bin₩mysqld.exe (mysqld 10.5.8-MariaDB) starting as process 24056 ...
Removing default user
Setting root password
Creating my.ini file
Registering service 'mariaDB'
Creation of the database was successful

C:₩Work₩mariadb-10.5.8-winx64>
```

$ docker build -t edowon0623/my_mariadb -f Dockerfile_mariadb .

$ docker run -d -p 3306:3306 **--network ecommerce-network** --name mariadb edowon0623/my_mariadb

# Run Kafka Server – 1)

- ***Zookeeper + Kafka Standalone***
  - docker-compose로 실행
  - git clone https://github.com/wurstmeister/kafka-docker
  - docker-compose-single-broker.yml 수정

$ docker-compose -f docker-compose-single-broker.yml up -d

```yaml
version: '2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    ports:
      - "2181:2181"
    networks:
      my-network:
        ipv4_address: 172.18.0.100
  kafka:
    # build: .
    image: wurstmeister/kafka
    ports:
      - "9092:9092"
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 172.18.0.101
      KAFKA_CREATE_TOPICS: "test:1:1"
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - zookeeper
    networks:
      my-network:
        ipv4_address: 172.18.0.101

networks:
  my-network:
    name: ecommerce-network
```

# *Run Kafka Server – 2)*

- bitnami/kafka
  - https://hub.docker.com/r/bitnami/kafka

```yaml
version: '2'

networks:
  my-network:
    name: ecommerce-network

services:
  zookeeper:
    image: 'bitnami/zookeeper:latest'
    ports:
      - "2181:2181"
    environment:
      ALLOW_ANONYMOUS_LOGIN: "yes"
    networks:
      my-network:
        ipv4_address: 172.18.0.100
  kafka:
    image: 'bitnami/kafka:latest'
    ports:
      - "9092:9092"
    environment:
      ALLOW_PLAINTEXT_LISTENER: "yes"
      KAFKA_CFG_ZOOKEEPER_CONNECT: "zookeeper:2181"
    depends_on:
      - zookeeper
    networks:
      my-network:
        ipv4_address: 172.18.0.101
```

# *Run Zipkin*

- https://zipkin.io/pages/quickstart

```
$ docker run -d -p 9411:9411 \
        --network ecommerce-network \
        --name zipkin \
        openzipkin/zipkin
```

# *Deployed Services – IP address, ports*

| Service (Container name) | IP address | Port |
|---|---|---|
| rabbitmq | 172.18.0.2/16 | 5671 |
| config-service | 172.18.0.3/16 | 8888 |
| discovery-service | 172.18.0.4/16 | 8761 |
| apigateway-service | 172.18.0.5/16 | 8000 |
| mariadb | 172.18.0.6/16 | 3306 |
| kafka-docker_zookeeper_1 | 172.18.0.100/16 | 2181 |
| kafka-docker_kafka_1 | 172.18.0.101/16 | 9092 |
| zipkin | 172.18.0.9/16 | 9411 |
| prometheus | 172.18.0.10/16 | 9090 |
| grafana | 172.18.0.11/16 | 3000 |
| user-service | - | - |
| order-service | - | - |
| catalog-service | - | - |

# Build, Run Users Microservice

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY target/user-service-1.0.jar UserService.jar
ENTRYPOINT ["java", "-jar", "UserService.jar"]
```

```
gateway:
  ip: 172.18.0.5
```

*apigateway-service*

- docker build --tag=edowon0623/user-service .
- docker push edowon0623/user-service

```
$ docker run -d --network ecommerce-network \
        --name user-service \
        -e "spring.cloud.config.uri=http://config-service:8888" \
        -e "spring.rabbitmq.host=rabbitmq" \
        -e "spring.zipkin.base-url=http://zipkin:9411" \
        -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" \
        -e "logging.file=/api-logs/users-ws.log" \
        edowon0623/user-service
```

# Build, Run Orders Microservice

```
public class KafkaProducerConfig {
    @Bean
    public ProducerFactory<String, String> producerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "127.0.0.1:9092");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);

        return new DefaultKafkaProducerFactory<>(props);
    }
```

*change the kafka broker*

| | | |
|---|---|---|
| kafka-docker_zookeeper_1 | 172.18.0.100/16 | 2181 |
| kafka-docker_kafka_1 | 172.18.0.101/16 | 9092 |

# Build, Run Orders Microservice

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY target/order-service-1.0.jar OrderService.jar
ENTRYPOINT ["java", "-jar", "OrderService.jar"]
```

- docker build --tag=edowon0623/order-service .
- docker push edowon0623/order-service

```
datasource:
#   driver-class-name: org.h2.Driver
#   url: jdbc:h2:mem:testdb
#   username: sa
#   password:
  driver-class-name: org.mariadb.jdbc.Driver
  url: jdbc:mariadb://localhost:13306/mydb
  username: root
  password: test1357
```

```
$ docker run -d --network ecommerce-network \
        --name order-service \
        -e "spring.zipkin.base-url=http://zipkin:9411" \
        -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" \
        -e "spring.datasource.url=jdbc:mariadb://mariadb:3306/mydb" \
        -e "logging.file=/api-logs/orders-ws.log" \
        edowon0623/order-service
```

# Build, Run Orders Microservice

- mariadb 접속 시 오류 발생 시

```
docker exec -it mariadb /bin/bash
root@54808b0489ab:/# mysql -h127.0.0.1 -uroot -p
Enter password:
ERROR 1130 (HY000): Host '127.0.0.1' is not allowed to connect to this MariaDB server
root@54808b0489ab:/#
```

```
root@54808b0489ab:/# mysql -uroot -p
Enter password:
```

```
MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> grant all privileges on *.* to 'root'@'%' identified by 'test1357';
Query OK, 0 rows affected (0.003 sec)

MariaDB [mysql]> flush privileges;
Query OK, 0 rows affected (0.003 sec)

MariaDB [mysql]> exit
```

# Build, Run Catalogs Microservice

```java
public class KafkaConsumerConfig {
    @Bean
    public ConsumerFactory<String, String> consumerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "127.0.0.1:9092");
        props.put(ConsumerConfig.GROUP_ID_CONFIG, "consumerGroupId");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);

        return new DefaultKafkaConsumerFactory<>(props);
    }
```

*change the kafka broker*

| | | |
|---|---|---|
| kafka-docker_zookeeper_1 | 172.18.0.100/16 | 2181 |
| kafka-docker_kafka_1 | 172.18.0.101/16 | 9092 |

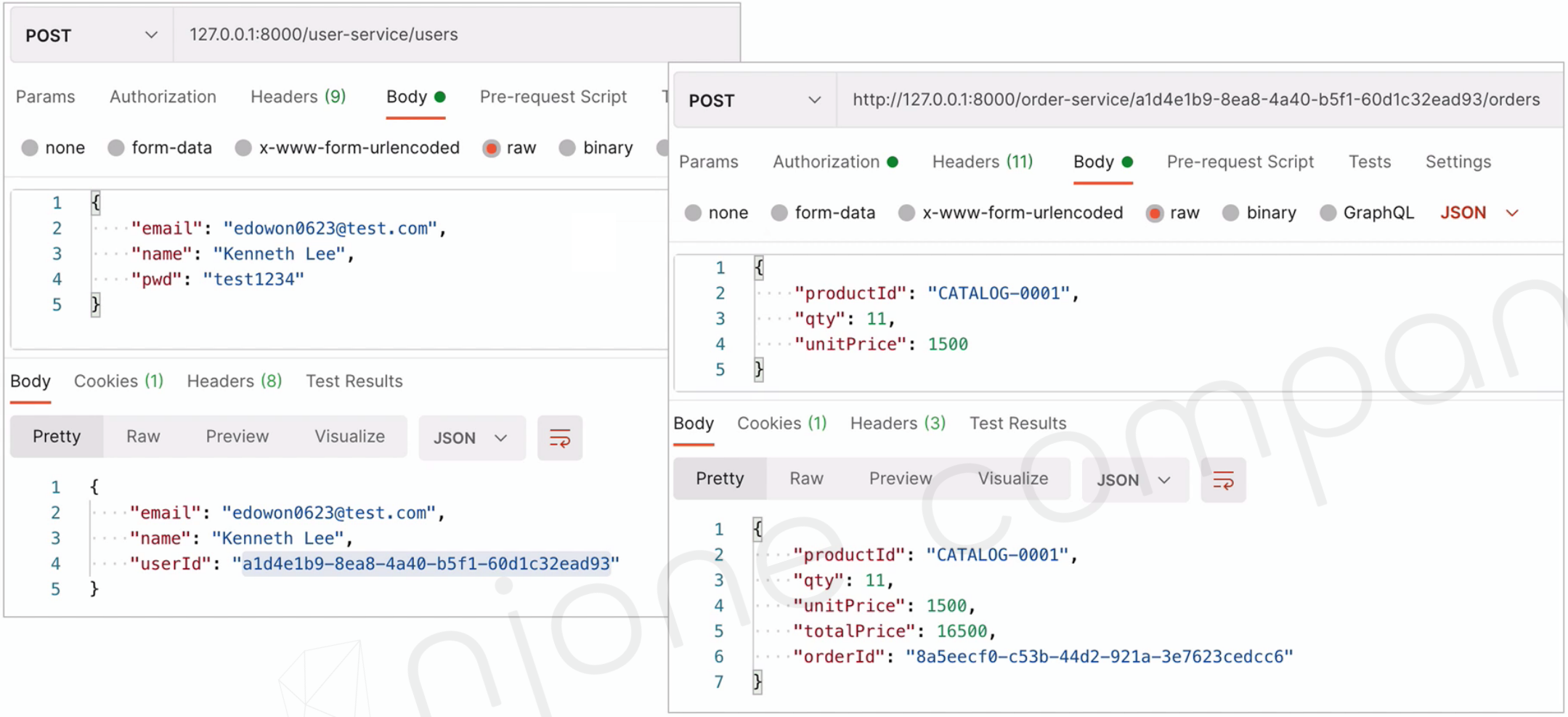
# Build, Run Catalogs Microservice

```
FROM openjdk:17-ea-11-jdk-slim
VOLUME /tmp
COPY target/catalog-service-1.0.jar CatalogService.jar
ENTRYPOINT ["java", "-jar", "CatalogService.jar"]
```

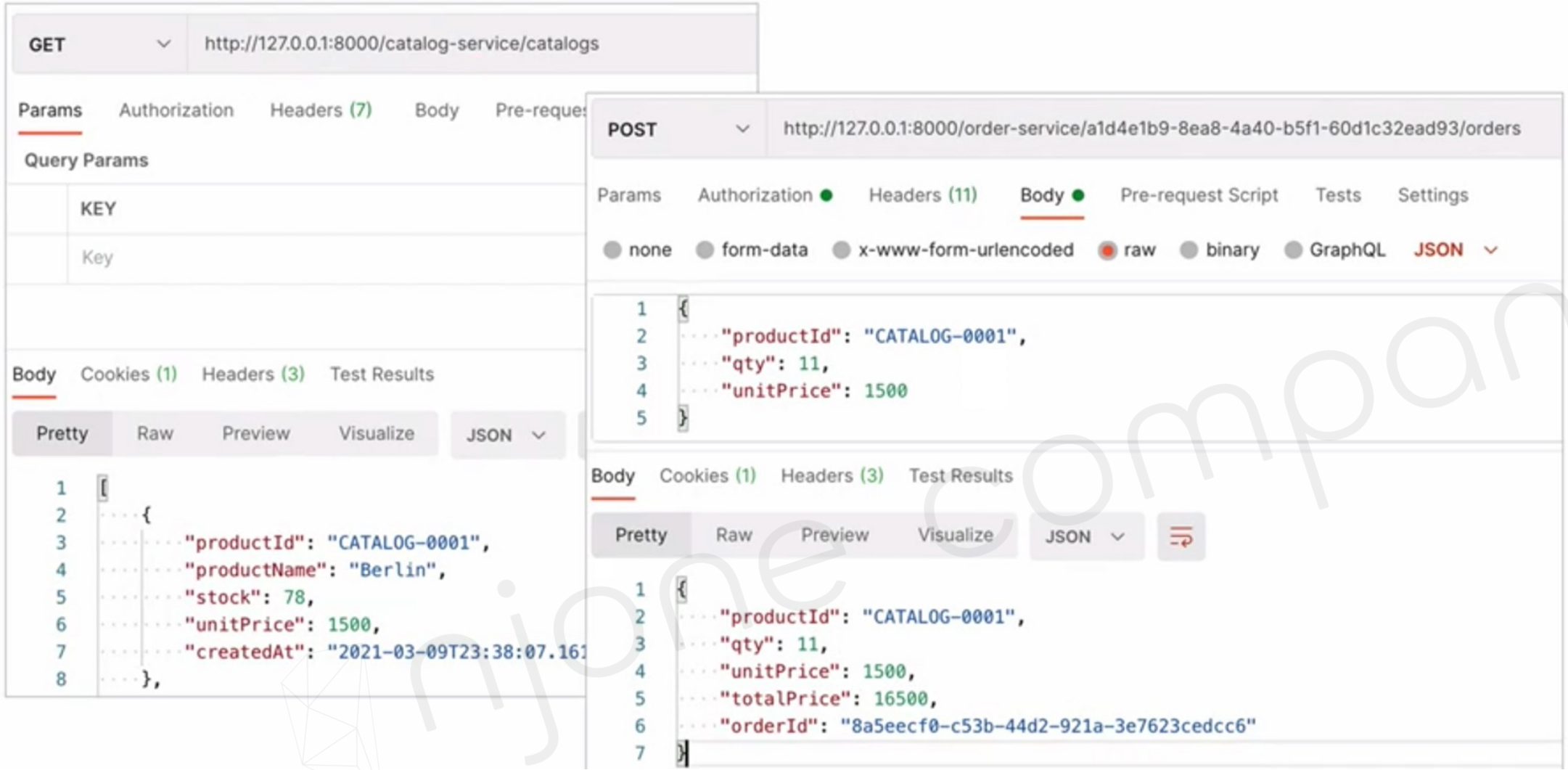
- docker build --tag=edowon0623/catalog-service .
- docker push edowon0623/catalog-service

```
$ docker run -d --network ecommerce-network \
        --name catalog-service \
        -e "eureka.client.serviceUrl.defaultZone=http://discovery-service:8761/eureka/" \
        -e "logging.file=/api-logs/catalogs-ws.log" \
        edowon0623/catalog-service
```

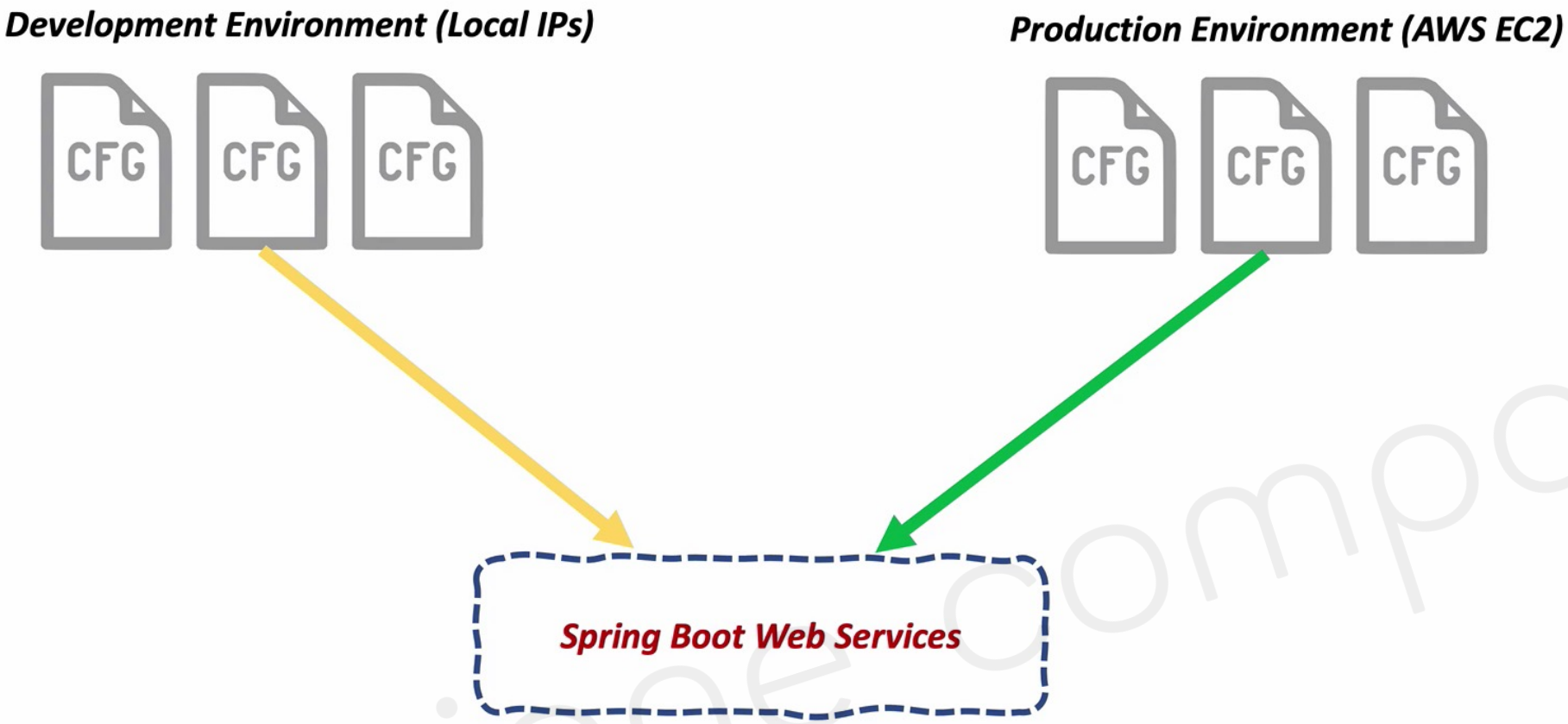
# Test


POST
127.0.0.1:8000/user-service/users
Params
Authorization
Headers (9)
Body
Pre-request Script
none
form-data
x-www-form-urlencoded
raw
binary
{
"email": "edowon0623@test.com",
"name": "Kenneth Lee",
"pwd": "test1234"
}
Body
Cookies (1)
Headers (8)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"email": "edowon0623@test.com",
"name": "Kenneth Lee",
"userId": "a1d4e1b9-8ea8-4a40-b5f1-60d1c32ead93"
}
POST
http://127.0.0.1:8000/order-service/a1d4e1b9-8ea8-4a40-b5f1-60d1c32ead93/orders
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"productId": "CATALOG-0001",
"qty": 11,
"unitPrice": 1500
}
Body
Cookies (1)
Headers (3)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"productId": "CATALOG-0001",
"qty": 11,
"unitPrice": 1500,
"totalPrice": 16500,
"orderId": "8a5eecf0-c53b-44d2-921a-3e7623cedcc6"
}

# Test


GET
http://127.0.0.1:8000/catalog-service/catalogs
Params
Authorization
Headers (7)
Body
Pre-reques
Query Params
KEY
Key
Body
Cookies (1)
Headers (3)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"productId": "CATALOG-0001",
"productName": "Berlin",
"stock": 78,
"unitPrice": 1500,
"createdAt": "2021-03-09T23:38:07.161
},
POST
http://127.0.0.1:8000/order-service/a1d4e1b9-8ea8-4a40-b5f1-60d1c32ead93/orders
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"productId": "CATALOG-0001",
"qty": 11,
"unitPrice": 1500
}
Body
Cookies (1)
Headers (3)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"productId": "CATALOG-0001",
"qty": 11,
"unitPrice": 1500,
"totalPrice": 16500,
"orderId": "8a5eecf0-c53b-44d2-921a-3e7623cedcc6"
}

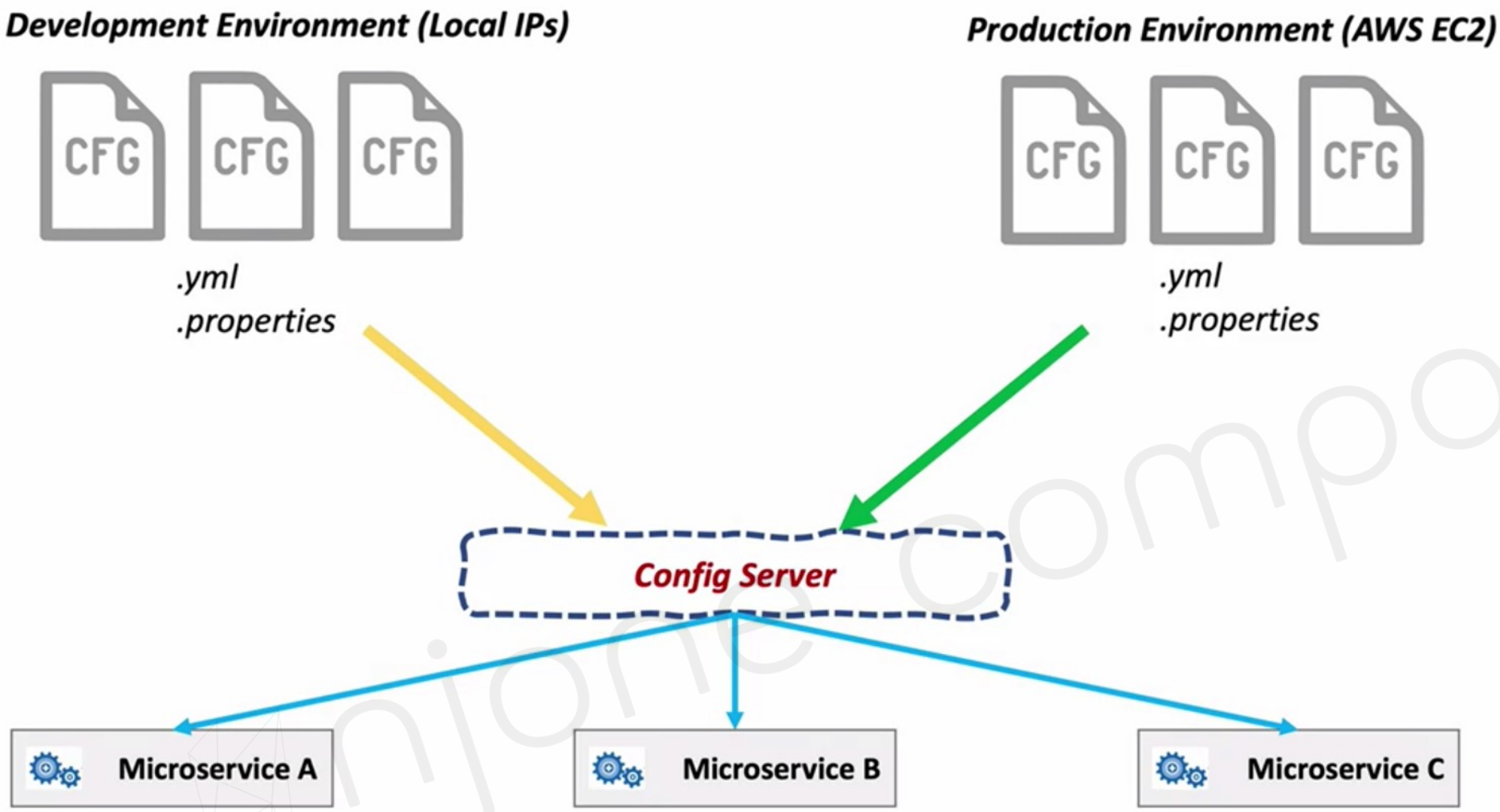
# Run Microservices




*$ mvn spring-boot:run –Dspring-boot.run.arguments=--spring.profiles.active=production*

*$ java –Dspring.profiles.active=default XXXXXXXX.jar*

# Run Microservices



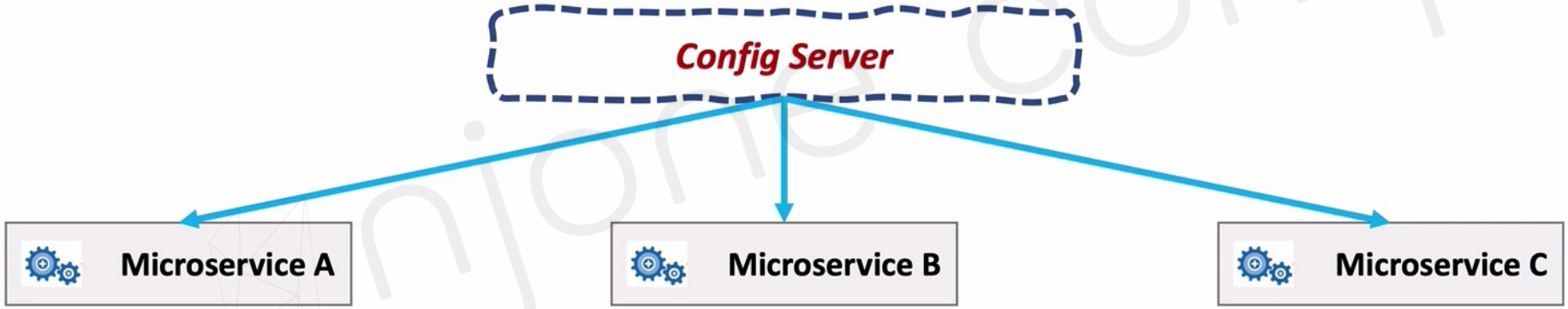
# Run Microservices

**Development Environment (Local IPs)**


*A.yml*
*B.yml*
*C.yml*

**Production Environment (AWS EC2)**


*A-<profile name>.yml*
*B-production.yml*
*C-production.yml*



# Run Microservices

```java
@Data
@Component
@Profile("dev")
public class MysqlConnector implements IConnector {
    public String getSource() {
        return "This is a mysql connector.";
    }
}
```

```java
@Data
@Component
@Profile("prod")
public class MongoDBConnector implements IConnector {
    public String getSource() {
        return "This is a MongoDB connector.";
    }
}
```

```java
System.out.println("Current connector:" + connector.getSource());
```

```
mvn spring-boot:run -Dspring-boot.run.arguments=--spring.profiles.active=dev
2021-03-08 23:06:32.360  INFO [user-service,c409196a2d419ad3,c409196a2d419ad3]
Current connector:This is a mysql connector.
```

```
mvn spring-boot:run -Dspring-boot.run.arguments=--spring.profiles.active=prod
2021-03-08 23:08:19.200  INFO [user-service,dc5198f94e5bd95c,dc5198f94e5bd95c]
Current connector:This is a MongoDB connector.
```